AF/2176 IFW

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231, on December 2, 2004.

Attorney for Applicant

PATENT
Docket No. ST9-99-146

Applicant:	Jan Burchhardt et al.	)
Serial No.:	09/587,581	)
Filed:	June 5, 2000	) Group Art Unit: 2176
For:	REPRESENTING IMS MESSAGES AS XML DOCUMENTS	)
Examiner:	Nathan Hillary	)

APPELLANTS' APPEAL BRIEF

Assistant Commissioner
For Patents
Alexandria, VA 22313-1450

Sir:

On October 4, 2004, Appellants filed a timely Notice of Appeal from the Final Office Action mailed August 2, 2004. Appellants appeal from the rejection of all pending claims.

This Brief is being filed under the provisions of 37 C.F.R. § 41.37. The filing fee set forth in 37 C.F.R. § 41.20(b)(2) of Three Hundred and Forty Dollars ($340.00) is now being submitted. The Commissioner is hereby authorized to charge payment of any additional fees associated with this communication, or to credit any overpayment, to Deposit Account No. 09-0460 .

12/07/2004 HGUTEMA1 00000004 090460 09587581
01 FC:1402 340.00 DA

1. REAL PARTY IN INTEREST

The real party in interest is the assignee, International Business Machines Corporation, Armonk, New York.

2. RELATED APPEALS AND INTERFERENCES

There are no related appeals and interferences.

3. STATUS OF CLAIMS

[0001] Claims 1-30 stand rejected under 35 U.S.C. §§ 112 and 103. The Final Office Action, mailed August 2, 2004, rejected claims 1-30 under 35 U.S.C. §112, second paragraph as being indefinite. Claims 1-5, 11-15, and 21-25 stand rejected under 35 U.S.C. §103(a) as obvious in view of U.S. Patent No. 5,715,453 to Stewart (hereinafter "Stewart"), U.S. Patent No. 5,737,597 to Blackman et al. (hereinafter "Blackman"), and a publication from W3C on WIDL (hereinafter "WIDL"). Claims 6, 16, and 26 stand rejected under 35 U.S.C. §103(a) as obvious in view of Stewart, Blackman, WIDL, and further in view of U.S. Patent No. 6,038,393 to Lyengar et al. (hereinafter "Lyengar"), and a publication on "XMI Opens Application Interchange" by Brodsky (hereinafter "Brodsky"). Claims 7-10, 17-20, and 27-30 stand rejected under 35 U.S.C. §103(a) as obvious in view of Stewart, Blackman, WIDL, and further in view of U.S. Patent No. 6,182,029 to Friedman et al. (hereinafter "Friedman").

4. STATUS OF AMENDMENTS

[0002] Appellants have not filed any Amendment subsequent to receipt of the Final Rejection.

5. SUMMARY OF CLAIMED SUBJECT MATTER

Claims 1, 11, and 21.

[0003] These claims define a method, system, and article of manufacture, respectively, that include substantially the same components. Claims 1, 11, and 21 teach representation of transaction processing system messages 30 in an open, interchangeable, format such as eXtensible Markup Language (XML) with newer computer systems. *Specification*, page 6, line 3, page 12, lines 6-9. In particular, the claims teach representation of input messages 30 and output messages 30 of applications operating on a transaction processing system, such as an Information Management System (IMS) from IBM. *Specification*, page 6, line 3, Figure 4. The input and output messages 30 represent input and output data between the application 18 on the transaction processing system and an interface such as a user terminal 28. *Specification*, Figure 4, page 4, lines 5-19.

[0004] The input and output messages 30 have a particular format defining the types and lengths of fields. This format is known as a message definition 38. *Specification*, page 4, line 21- page 5, line 2. The message definitions 38 are defined in source code of various computer programming languages including COBOL, Assembler, PL/I, Pascal, and the like. *Specification*, page 5, lines 2-5.

[0005] Claims 1, 11, and 21 teach conversion of proprietary message definitions 38 and input/output messages 30 into an open and interchangeable format, XML document 44. *Specification*, Figure 7. In one embodiment, an XML document template 58 is generated from a transaction processing system (IMS) message definition 38. *Specification*, page 21, lines 11-16, Figure 7. Then, a transaction processing system (IMS) message 30 is merged with the template 58 to produce an XML document 44. *Id.* These claims also teach that the message definition 38 represents the "syntax and semantics" for the messages. *Specification*, page 12, lines 10-12. As explained below, the syntax and semantics allows for proper parsing and transformation of the message definitions 38 and hence the messages 30.

Claims 2 and 3.

[0006] These claims obtain an IMS message definition 38 and a Document Type Definition (DTD) 54 associated with an arbitrary IMS message definition 38 to facilitate generation of the XML document template 58, the present invention may. *Specification*, page 19, lines 13-19, page 21, lines 11-16, Figure 7. In Claim 2, the IMS message definition 38 is compiled by a compiler 76 to produce an associated data (Adata) file 78. *Specification*, page 13, lines 3-8. The Adata file 78 is then parsed using the DTD 54 to produce the XML document template 58. *Specification*, page 25, lines 6-12. In Claim 3, the message definition 38 is parsed using the DTD 54 to produce a corresponding XML document template 58. *Specification*, page 25, lines 13-15.

Claim 4.

[0007] Claim 4 teaches that the Adata file 78 includes a message definition 38 in a format substantially semantically equivalent to the program source code 74 from which the message definition 38 originates. *Specification*, page 13, lines 9-10 and 20-21. In particular, the symbol record contains information related to the message definition 38. *Specification*, page 16, lines 5-9, Table 3. In other words, the Adata file 78 and source code 74 contain the information that means substantially the same thing. *Specification*, page 17, lines 18-20.

Claim 5.

[0008] Claim 5 teaches extraction of a message definition 38 from a source code file 74 and/or a copy file. *Specification*, page 18, lines 13-16, Figure 7. A message definition extractor 75 may be provided for this purpose. *Specification*, page 18, lines 16-18, Figure 7.

Claim 6.

[0009] Claim 6 teaches creation of a Uniform Modeling Language (UML) object model 64 that represents arbitrary message definitions 38 using a UML modeling tool 62. *Specification*, page 22, lines 1-2, lines 8-9, Figure 7. One example model 64 is illustrated and described in relation to Figure 8. *Specification*, Figure 8, page 22, lines 9-19. The DTD 54 is generated from the UML object model 62. *Specification*, page 22, lines 8-9. The UML object model 64 is then

processed using an XMI utility 66 to generate the DTD 54. *Specification*, page 22, line 21 – page 23, line 7, Figure 7. Preferably, the DTD 54 is a generic DTD 54 that may be used for all messages 30. *Specification*, page 20, lines 19-21.

Claim 7.

[0010] Claim 7 teaches merging of the message 30 into the XML document template 58 to produce the XML document 44. Initially, a placeholder, such as dataitem 55, is identified in the XML document template 58. *Specification*, page 26, lines 19-20, Figure 13. Provided the placeholder has a value, a value corresponding to the placeholder, dataitem 55, is then read from the message 30. *Specification*, page 26, line 20 – page 27, line 2, page 27, lines 3-5, Figure 13. Next, the value is inserted into XML document template 58 in the location indicated by the placeholder (*i.e.* dataitem 55). *Specification*, page 27, lines 7-9. Once all values are inserted the XML document template 58 has been transformed into the XML document 44.

[0011] The present invention allows interfacing of a transaction processing system, such as IMS (Information Management System), with newer computer systems such as Internet servers, web clients, network clients, and any other interface configured to interpret and produce XML messages. The present invention properly converts between IMS message definitions and XML documents representing IMS message definitions.

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The following grounds of rejection are presented for review on appeal:

I. The rejection under 35 U.S.C. §112, second paragraph of Claims 1-30 as indefinite.

II. The rejection under 35 U.S.C. §103(a) of Claims 1-5, 11-15, and 21-25 as obvious in view of Stewart, Blackman, and WIDL.

III. The rejection under 35 U.S.C. §103(a) of Claims 6, 16, and 26 as obvious in view of Stewart, Blackman, WIDL, Lyengar, and Brodsky.

IV. The rejection under 35 U.S.C. §103(a) of Claims 7-10, 17-20, and 27-30 as obvious in view of Stewart, Blackman, WIDL, and Friedman.

7. ARGUMENT

I. The rejection under 35 U.S.C. §112, second paragraph of Claims 1-30 as indefinite.

[0012] Standard for indefiniteness. 35 U.S.C. §112, 2nd paragraph requires that Appellants particularly point out and distinctly claim the subject matter the Appellants regard as their invention. This determination is to be made "not in a vacuum, but in light of: (A) The content of the particular application disclosure; (B) The teachings of the prior art; and (C) The claim interpretation that would be given by one possessing the ordinary level of skill in the pertinent art at the time the invention was made." See MPEP §2173.02.

Claims 1, 11, and 21.

[0013] Claims 1, 11, and 21 were rejected relating to clarity of the phrase "representative of the syntax and semantics." The Examiner provides no further indication as to what is unclear. Appellants respectfully assert that this phrase is sufficiently definite when read in light of the specification, teachings in the art, and the interpretation of one of skill in the art would give the phrases.

[0014] The phrase "representative of the syntax and semantics" was recently added to Claims 1, 11, and 21 to clarify the meaning of a transaction processing system message definition. The phrase is clearly understood by those of skill in the art because the terms "syntax" and "semantics" are well known terms.

[0015] Those of skill in the art are computer science graduates and other computer programmers with equivalent experience in computer programming technologies including XML, XMI, COBOL, C, C++, Pascal, Assembler, programming languages in general, and transaction processing systems, such as IMS. Such individuals learn early on the meaning and difference between the terms syntax and semantics. As used in the art, the term "syntax" refers

to the format, layout, or structure of a data within some data structure such as a document, record, array, object, or the like.

[0016] A common resource for technical definitions defines "syntax" as "the spelling and grammar of a programming language. Computers are inflexible machines that understand what you type only if you type it in the exact form that the computer expects. The expected form is called the syntax. Each program defines its own syntactical rules that control which words the computer understands, which combinations of words are meaningful, and what punctuation is necessary." *See* definition of 'syntax' on www.webopedia.com, attached for convenience.

[0017] In the claimed invention, "syntax" of a message definition is described in relation to a COBOL or other source code definition of a message. *See* specification page 12, lines 10-12. Appellants note that here, references are made to the "format" and the "syntax" of a message definition. Such information needs to be included in a transaction processing message definition representative of the source code message definition so that the sender and receiver of messages based on the transaction processing message definition both understand the data.

[0018] Similarly, the term "semantics" is well known. Semantics refers to the meaning of a data item. The semantics are what distinguish a data item such as "1" as being a number versus a representation of the Boolean value "True." Again the definition of the term from the website webopedia is instructive. "In computer science, the term "semantics" is frequently used to differentiate the meaning of an instruction from its format. The format, which covers the spelling of language components and the rules controlling how components are combined, is called the language's syntax." *See* definition of "semantics" on www.webopedia.com, attached for convenience.

[0019] The message definition defines the message. Because the sending system and receiving system both need to understand the format (syntax) and meaning (semantics) of the data in a message, this information needs to be included in the message definition. Therefore, Appellants submit that use of the phrase "representative of the syntax and semantics" in relation to a transaction processing system message definition in Claims 1, 11, and 21 is sufficiently clear and definite to those of skill in the art.

Claims 2 and 3.

[0020] Claims 2 and 3 depend from Claim 1 and include none of the particular language the Examiner has indicated as indefinite. Therefore, Appellants submit that these claims are sufficiently definite for the same reasons set forth above in relation to Claim 1, 11, and 21.

Claim 4.

[0021] Claim 4 was rejected relating to clarity of the phrase "substantially semantically equivalent to." The Examiner provides no further indication as to what is unclear. Appellants respectfully assert that this phrase is sufficiently definite when read in light of the specification, teachings in the art, and the interpretation of one of skill in the art would give the phrases.

[0022] The phrase "substantially semantically equivalent to" in Claim 4 relates to representations of transaction processing system message definitions in an Adata file. The phrase was added to clarify the phrase "relatively language independent format." The intent of the amendment is to indicate that the "meaning" or "semantics" of the message definition is substantially the same as that of message definitions found in source code.

[0023] Based on the definition of semantics provided above, Appellants respectfully submit that the phrase "substantially semantically equivalent to" is clear to one of skill in the art. A computer scientist reviewing this phrase within the context of the specification and the claims would understand that there is very little difference in meaning between a message definition in the Adata file and a message definition in a set of source code. This understanding is further supported in the specification on page 17, lines 19-20.

[0024] Therefore, Appellants respectfully assert that the phrases "representative of the syntax and semantics" and "substantially semantically equivalent to" as found in representative Claims 1 and 4 are clear and definite. This is particularly the case when examined in view of the specification, the art, and the interpretation that would be given by those of skill in this art, computer scientists. Appellants respectfully request that this rejection be reversed.

Claim 5.

[0025] Claim 5 depends from Claim 1 and includes none of the particular language the Examiner has indicated as indefinite. Therefore, Appellants submit that this claim is sufficiently definite for the same reasons set forth above in relation to Claim 1, 11, and 21.

Claim 6.

[0026] Claim 6 depends from Claim 1 and includes none of the particular language the Examiner has indicated as indefinite. Therefore, Appellants submit that this claim is sufficiently definite for the same reasons set forth above in relation to Claim 1, 11, and 21.

Claim 7.

[0027] Claim 7 depends from Claim 1 and includes none of the particular language the Examiner has indicated as indefinite. Therefore, Appellants submit that this claim is sufficiently definite for the same reasons set forth above in relation to Claim 1, 11, and 21.

II. The rejection under 35 U.S.C. §103(a) of Claims 1-5, 11-15, and 21-25 as obvious in view of Stewart, Blackman, and WIDL.

[0028] The Prior Art. The three references combined to reject the claims under Section 103 are summarized below.

[0029] Stewart. Stewart discloses a method that enables a web server to properly retrieve dynamic data results for web pages. Function calls within the web pages that query a data source for dynamic data are sent to one of a plurality of language processors as indicated by a configuration file. *Stewart*, Abstract and Figure 1. The appropriate language processor receives the function call, interacts with the data source to obtain query results, and returns the query results to a transaction processor. *Id.* The transaction processor then puts the query results into a web page which is sent to a user. *Id.*

[0030] Thus, Stewart addresses the problem of adapting a transaction processor using software changes to accommodate dynamic data function calls that access different data sources or different types of data sources. *Stewart*, Col. 1, lines 59-63. Stewart allows the transaction processor to remain unchanged while suitable language processors are included to accommodate different types of dynamic data function calls.

[0031] In Stewart, rather than one or more language processors bundled within the transaction processor, the transaction processor appropriately directs different types of dynamic data function calls to separate language processors to improve flexibility and minimize software changes to the transaction processor. Thus, the invention taught by Stewart allows a web server to serve up web pages having dynamic data results from a plurality of data sources which may change without requiring a corresponding software change to the web server or its transaction processor.

[0032] Blackman. Blackman teaches building object oriented software to interconnect object oriented applications with non-object oriented datastores such as an IMS DBMS from IBM. *Blackman*, Abstract. Blackman teaches persistent object-oriented software to access the IMS data in IMS datastores. Blackman teaches software for accessing IMS data. The software in Blackman is new middleware relating to fields and rows of a database, not format and meaning of messages.

[0033] WIDL. WIDL teaches a Web Interface Definition Language. The WIDL reference teaches an implementation of XML for defining an interface having services and bindings. *See* WIDL section 5.1. This means that all WIDL files conform to the rules and formatting requirements defined in XML. The WIDL defines an interface that enables non-web-based applications to access web resources. The web resource may be a web page or a functional module. Consequently, the WIDL defines a method that an application can call to get a specific service performed by the web service URL. *See* WIDL section 5.2, service attribute. The method is called by activating the web service URL.

[0034] Prima Facie Obviousness under 35 U.S.C. § 103. "To establish *prima facie* obviousness of a claimed invention, **all the claim limitations** must be taught or suggested by the prior art." MPEP §2143.03 (emphasis added). The Federal Circuit has held that "the 'subject matter' that must have been obvious to deny patentability under §103 is the entirety of the claimed invention," *Panduit Corp. v. Dennison Mfg. Co.*, 810 F.2d 1561, 1576 (Fed. Cir. 1987). In the present case however, Stewart, Blackman, and WIDL fail to disclose the entirety of the claimed invention.

[0035] Specifically, Stewart, Blackman, and WIDL fail to disclose a transaction processing system message definition. This message definition and the associated message are used throughout the independent claim elements. Consequently, Stewart, Blackman, and WIDL also fail to teach or disclose the remaining elements of the independent claims as they relate to a message definition and message.

Claims 1, 11, and 21.

[0036] Stewart, Blackman, and WIDL fail to disclose "generating an XML document template from **a transaction processing system message definition**..." (emphasis added) as recited in Claim 1 of the present invention. *Claims*, claim 1. Similar language is found in Claims 11 and 21. Specifically, these references fail to disclose or suggest a transaction processing system message definition.

[0037] Transaction processing system message definitions are specific data structures that identify a transaction processing application to service a transaction processing system message, and define various fields for transaction processing system messages sent to a specific transaction processing application. *See* specification page 4, line 20 - page 5, line 5. The transaction processing application is identified so that the transaction processing system can schedule processing of a transaction relating to the transaction processing system message. The transaction processing system message definitions are used to define input and output messaging with the transaction processing application. This messaging may have nothing to do with the database records.

[0038] The transaction processing system message definitions are defined when the transaction processing application is written (source code is created) to enable terminals and other devices to interact with the transaction processing application. *See* specification page 4, line 20 - page 5, line 5. Transaction processing system messages may or may not result in the transaction processing application accessing data such as IMS data of a transaction processing system. The message definitions enable messaging (*i.e.,* transaction communication) between the transaction processing system and a remote, external, terminal or client software module.

[0039] In order to communicate, the client or terminal and the transaction processing system must both know the format (syntax) and meaning (semantics) of data within the messages. As discussed above, the transaction processing system message definition provides this information such that an XML template can be generated and messages can be merged with the XML template into an XML file as recited in Claims 1, 11, and 21. The XML file comprises the common, open, interoperable message format used by both the transaction processing system and the client or terminal. Of course, the client software module must also be configured to read and interpret the XML file.

[0040] The Examiner suggests that Stewart, Blackman, and WIDL combined disclose "generating an XML document template from **a transaction processing system message definition**..." *Office Action*, August 2, 2004, page 4. However, the Examiner provides no indication as to what parts of Stewart disclose the particular parts of Claims 1, 11, or 21. The Examiner cites a lengthy discussion in Stewart relating to handling of web page requests in which the web page includes dynamic data. But, the Examiner still does not indicate what elements in Stewart teach corresponding elements of Claim 1 such as "an XML document template" and "a transaction processing system message definition."

[0041] Case law clearly establishes that "[t]he examiner cannot sit mum, leaving the applicant to shoot arrows into the dark hoping to somehow hit a secret objection harbored by the examiner. [The examiner] must state clearly and specifically any objections. . . and give the applicant fair opportunity to meet those objections with evidence and argument." In re Oetiker, 24 U.S.P.Q.2d 1443, 1447 (Fed. Cir 1992) (Plager, J. concurring). However, here the Examiner is simply citing a bulk of text which seems to have some tenuous relationship to the claimed

technology with no clear or specific indication as to what elements in Stewart teaches particular elements of Claim 1. This practice is costly in terms of time and expense for the Appellants and contrary to the spirit of cooperation that should prevail in working with the Examiner.

[0042] Appellants find no teaching or suggestion in Stewart of an XML document template, a transaction processing system message definition, or a transaction processing system message. Therefore, absent evidence to the contrary, Appellants respectfully submit that Stewart fails to teach or disclose all the elements of independent Claims 1, 11, and 21.

Stewart analysis regarding Claim 1

[0043] Stewart does teach use of macro files to produce the appropriate HTML in response to a web page request. *Stewart* Col. 4, lines 24-25. Stewart clearly teaches receipt of a web page request which is well known to be a message rather than a document. *Stewart* Figure 1. Stewart further discloses reading of a configuration file and processing of an HTML page. *Stewart* Col. 4, lines 25-28. However, the message passing in Stewart is between the web server application and the web browser. *Stewart* Figure 1.

[0044] Furthermore, there is no teaching or suggestion of a transaction processing system message definition. In Stewart, the messaging is between the web browser and the web server. There is no need to define a message definition because a standard message protocol is being used. Specifically, in Stewart the messaging protocol is the Hyper-Text Transfer Protocol (HTTP) which is commonly used to request and receive HTML pages. *Stewart* col. 6 lines 22-26. In contrast, the claimed invention includes a messaging definition to facilitate interfacing a client that uses open standards such as XML with a legacy transaction processing system such as IMS. A messaging definition is required because the format and semantics of messages may vary widely and this interpretation. This difference was clarified with the language added to Claim 1 indicating that the transaction processing system message definition is "representative of the syntax and **semantics** for messages exchanged with the transaction processing system." *See* Claim 1 (emphasis added). Arguably, an HTML file and/or macro file may represent the format or from one transaction processing system application to the next.

[0045] Under an unreasonably broad reading of Stewart, the HTML pages and/or macro files may be construed to read on a transaction processing system message definition. Appellants

disagree with such an interpretation. In an unreasonably broad sense, one may consider that these files define what will be sent between the web server and the web browser. However, there is a clear difference between what is claimed, the syntax and semantics of a message, and an HTML page. The HTML file and/or macro files clearly do not include any representation for the meaning (semantics) of the data in the file. The HTML and macro files contain information representing the presentation of data in the file. The meaning of the data is important for proper message passing as explained above. Therefore, Appellants find no teaching in Stewart of an XML document template, a transaction processing system message definition, or transaction processing system message.

[0046] Finally, Appellants also find no teaching in Stewart of "merging a transaction processing system message with the generated template to produce a corresponding XML document." *See* Claim 1. Stewart fails to teach or disclose a transaction processing system message. Appellants find nothing in Stewart that can be construed as a transaction processing system message. A transaction processing system message is a message that conforms to the transaction processing system message definition. *See* specification page 4, line 20 - page 5, line 5. In other words, data in the transaction processing system message is formatted (syntax) and has a meaning (semantics) as set forth in the transaction processing system message definition.

[0047] Appellants respectfully assert that Stewart fails to teach, disclose, or suggest generating an XML document template, a transaction processing system message definition, a transaction processing system message, or merging of a transaction processing system message with an XML document template to produce an XML document.

Blackman analysis relating to Claim 1.

[0048] Blackman simply teaches reading of a record layout from a COBOL "copylib." Reading the Blackman teaching in context, the record layout is for data fields in a data store or database record, not message fields in a message. *Blackman* col. 8, lines 38-52. Those of skill in the art recognize the difference between an interprocess message and a database record. The differences depend on how the messages are used. For example, input messages may include fields that are never stored in a data store record. Furthermore, an output message may include informational fields that provide information that originates from an application. However, the

data in the informational fields may never be stored in a data store record. In contrast, fields in a database record relate solely to data that is stored in the database. Therefore, the Blackman reference does nothing to cure the deficiencies of Stewart.

WIDL analysis relating to Claim 1.

[0049] Similarly, WIDL fails to teach all the elements of Claim 1 or provide teachings lacking in Stewart or Blackman. Specifically, Claim 1 recites "generating an XML document template...[and]...produc[ing] a corresponding XML document." WIDL teaches at most XML documents. There is no teaching in WIDL about an XML document template. WIDL also fails to teach or disclose generating an XML document template from a transaction processing system message definition or merging a transaction processing system message with the template to produce the XML document.

Give all Claim limitations proper weight

[0050] In the present case, Appellants amended Claims 1, 11, and 21 to clarify what is meant by a transaction processing system message definition. Specifically, the Appellants added the phrase "representative of the syntax and semantics" to Claim 1, 11, and 21. Appellants respectfully assert that the Examiner failed to give proper consideration to the added clarifying phrase. Therefore, the Examiner failed to consider the "entirety of the claimed invention."

[0051] "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). *MPEP* §2143.03. Although the meaning of the phrases were not clear to the Examiner, which prompted the rejection under 35 U.S.C §112, 2nd paragraph, the clarifying language clearly distinguishes the claims from the art of record.

[0052] For reasons explained above, those of skill in the art will recognize that the transaction processing system message definition defines the syntax and semantics. Furthermore, the meaning (semantics) of data in the messages that are exchanged is critical to successful messaging between the client and client terminals and the transaction processing system such as IMS. As indicated above, these amendments are supported in the specification and should have

been considered in determining the obviousness of these claims in relation to the prior art of record.

[0053] Appellants respectfully assert that this oversight by the Examiner has cost the Appellants unnecessary delay and expense in procuring a patent for the claimed invention. As the features included in the amended statement bolster Appellants' assertion that Claims 1, 11, and 21 are nonobvious. Appellants respectfully request that the rejection be overruled.

[0054] Neither, Stewart, Blackman, nor WIDL, singly or in combination, teach the entirety of the limitations of the present invention. Specifically, there is no teaching of a transaction processing system message definition, a transaction processing system message, and an XML template. Furthermore, these references fail to teach or disclose all the operations that are performed in Claims 1, 11, and 21 with respect to the transaction processing system message definition (i.e. generating an XML document template using the transaction processing system message definition). Because these limitations render the present invention unobvious, Appellants assert that Claims 1, 11, and 21 are allowable.

Teaching to combine.

[0055] *Prima Facie* Obviousness. The Federal Circuit has recently stated: [A] rejection cannot be predicated on the mere identification ... of individual components of claimed limitations. Rather, particular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed. *In re Kotzab*, 217 F.3d 1365, 1371, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000). Appellants assert that the Examiner has failed to provide objective evidence from the prior art why one would make the combination suggested by the Examiner. Instead, Appellants respectfully submit that the motivation for the combination has been improperly derived from the Appellants' disclosure. It is "impermissible to use the claims as a frame and the prior art references as a mosaic to piece together a facsimile of the claimed invention." *Uniroyal v. Rudkin-Wiley*, 5 USPQ2d 1434, 1438 (Fed. Cir. 1988) (citing *W. L. Gore & Associates v. Garlock, Inc.*, 220 USPQ 303, 312). Yet, that is what the Examiner has done; citing references

with certain elements of the present invention but lacking all limitations of the present invention or any suggestion or teaching for the combination.

[0056] Furthermore, "[an] examiner's conclusory statements [the court quotes the conclusory statements] do not adequately address the issue of motivation to combine. This factual question of motivation is material to patentability..." *In re Lee*, 277 F.3d 1338, 61 USPQ2d 1430 (Fed. Cir. 2002). Appellants assert that the Examiner failed to provide objective evidence why Stewart, Blackman, WIDL, Lyengar, Brodsky, and Friedman would be selected and why these references would be combined by one of ordinary skill in the art.

[0057] Appellants have argued above that Stewart, Blackman, WIDL, Lyengar, Brodsky, and Friedman do not teach or disclose a transaction processing system message definition as well as other claimed features. However, even if Stewart, Blackman, WIDL, Lyengar, Brodsky, and Friedman did teach a transaction processing system message definition as recited in the claims, Appellants find no motivation in **the references** to combine Stewart, Blackman, WIDL, Lyengar, Brodsky, and Friedman or the desirability of such a combination to arrive at the present invention.

[0058] The Examiner cites to rationale such as "defin[ing] and generat[ing] code for datastore persistent objects...and WIDL definitions [which] provide a mapping between web resources and applications written in conventional programming languages..." *See* Final Office Action, page 9. However, such an objective is not desirable in relation to the claimed invention. The claimed invention seeks to represent transaction processing system messages as XML documents.

[0059] Appellants fail to see, and asserts that others skilled in the art would fail to see, the connection between the desirability of the combination related by the Examiner and the purpose recited in the preambles of the claimed invention. The claimed invention does not generate code. Instead, transaction processing system message definitions, XML document templates, and transaction processing system messages are used to generate an open interchangeable XML document that may be used in messaging between modern clients and a legacy transaction processing system such as IMS. The cited Stewart, Blackman, WIDL,

Lyengar, Brodsky, and Friedman fail to disclose, suggest, or motivate one to modify the primary reference Stewart to arrive at the present invention ***as a whole***.

[0060] Appellants respectfully assert that **if** the prior art of record so clearly demonstrates the obviousness of the claimed invention, a single reference would teach more than just one or two elements of the claimed invention. However, the formation of the combinations used in the rejections is indicative of impermissible hindsight analysis by the Examiner. The sheer number of references used indicates that the claim terms were used in a keyword search of the prior art. For certain claims, up to six different references are relied upon. Once a keyword hit was found, there appears to be little analysis performed to determine the applicability of relevance of the reference. The large set of text in Stewart cited in rejecting Claim 1 without any further explanation as to which claim elements are represented by which elements in Stewart is indicative of hindsight keyword analysis in support of a §103 obviousness rejection. Appellants respectfully assert that because such analysis is improper the rejections should be overturned.

[0061] Appellants respectfully assert that the teaching or suggestion to make the claimed combination is only found in the Appellant's disclosure. Neither, Stewart, Blackman, nor WIDL teach "generating an XML document template from a transaction processing system message definition, the message definition representative of the syntax and semantics for messages exchanged with the transaction processing system." The present invention provides the only teaching for this element. Appellants assert that Claims 1, 11, and 21 are allowable because the teaching to generate an XML document template from a transaction processing system message definition is only found in the present invention and not in the prior art relied on in the rejection.

[0062] The Examiner rejected Claims 1, 11, and 21 by picking and choosing isolated teachings from the Stewart, Blackman, WIDL, Lyengar, Brodsky, and Friedman references and pasting them together to recreate the claims. None of these references suggest nor teach the combination of the present invention, "generating an XML document template from a transaction processing system message definition, the message definition representative of the syntax and semantics for messages exchanged with the transaction processing system." *Claims*, Claim 1. Because the Examiner has shown no motivation or suggestion to combine the prior art references

and the desirability of such a combination conflicts with that stated in the claim preamble. Appellants assert that Claims 1, 11, and 21 are allowable.

Claims 2, 3.

[0063] Claims 2 and 3 depends from Claim 1 and should be allowed for all the same reasons stated above regarding Claim 1. In addition, claim 2 includes the limitation "...obtaining a DTD for representing arbitrary transaction processing system message definitions...compiling the transaction processing system message definition...to produce an Adata file; and parsing the Adata file using the DTD to generate an XML document template." *Claims*, Claim 2. Claim 3 includes very similar limitations. Appellants submit that the prior art of record fails to teach the above recited elements.

[0064] The Examiner again recites large sections of text in the prior art references and then makes no indication as to how each reference teaches each element or even each step recited in Claim 2. *Office Action*, August 2, 2004, page 6. The Examiner acknowledges that these particular features of Claim 2 (quoted above) are not explicitly taught in the references.

[0065] However, the Examiner goes on to make a broad, unsupported, assertion that those in the art would interpret WIDL and Blackman **as having the capabilities** quoted above. *Office Action*, August 2, 2004, page 6. Appellants respectfully disagree because Appellants find nothing in WIDL or Blackman that hints at the elements of Claim 2. The assertion by the Examiner begs the question, why would one of skill in the art believe these references **have the capabilities** recited in Claim 2, when these references fail to even use the same terms? Appellants assert that the computer science art is a highly particular scientific field that uses very particular terminology to represent concepts that may have subtle distinctions but are different none the less. This is clearly evident in the vast number of abbreviations and acronyms used in the art. Therefore, if those of skill in the art would recognize that the references would obviously have the capabilities of Claim 2, Appellants assert that the references would include the same terminology and/or acronyms, such as Adata and DTD.

[0066] Next, the Examiner refers to WIDL, webopedia.com, and a Hutchinson Dictionary to assert that the elements and steps of Claim 2 combined with Stewart are obvious to one of skill

in the art. *Office Action*, August 2, 2004, page 8. The Examiner suggests that those in the art would know that COBOL must be compiled and that a DTD must be used to render the COBOL in XML. Regardless of whether the Examiner's assertion is true, there are still at least three elements that are missing, namely, a transaction processing system message definition, producing of an Adata file, and parsing of the Adata file.

[0067] Appellants find nothing in WIDL and/or Blackman to suggest "a DTD for representing **arbitrary** transaction processing system message definitions." As discussed above, there is no teaching of any structures that correspond to a transaction processing system message definition. Therefore, even if WIDL included the capability to express a DTD for representing arbitrary transaction processing system message definitions there is no teaching to do so because Stewart, Blackman, and WIDL have no teaching to define an arbitrary transaction processing system message definition. Furthermore, there is no teaching of a Document Data Type (DTD) relating to an arbitrary transaction processing system message definition.

[0068] Furthermore, Appellants find no references to the term "Adata file" in the art of record and no indication as to its obvious equivalent. Appellants submit that "[a]ll words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). *MPEP* §2143.03. The term "Adata" is very particular, not misspelled, and represents a permissible name used in trade. *MPEP* §608.01(v). The Adata file is a file produced by a compiler of programming source code as a debugging aide. *See* specification page 13, lines 3-21 generally, and lines 6-8 in particular. The Examiner has not proposed any other files in the art that are comparable to an Adata file. Appellants have not been able to identify any files in the art of record produced by a source code compiler that teach or suggest an Adata file. Appellants assert that including a limitation to such a specific type of file (Adata file) alone should distinguish Claim 2 as unobvious over the art of record.

[0069] Appellants respectfully submit that the Examiner failed to give proper weight and consideration to the terms in Claims 2 and 3. While the exact terms need not be found in the references, there must be either clear elements in the references that teach the claim terms or a clear indication from the Examiner as to what is considered as teaching **ALL** the claim terms.

Appellants submit that, for the reasons set forth above, the Examiner has done neither. Appellants submit that such failure by the Examiner is reversible error.

Claim 4.

[0070] Claim 4 depends from Claim 2 and should be allowed for all the same reasons stated above regarding Claim 2. In addition, claim 4 adds the limitation "...the Adata file comprises a transaction processing system message definition in a format substantially semantically equivalent to program source code..." *Claims*, 4. This claim recites the features that a transaction processing system message definition in the Adata file includes substantially the same meaning (semantics) as the original transaction processing system message definition found in the programming source code. *See* specification page 17, lines 18-20. The benefit of this is that one parser that can interpret an Adata file can be written and used instead of a separate parser for each particular programming language containing transaction processing system message definitions. *See* specification page 13, lines 9-10. This is a significant savings of time and programming effort which makes generating the XML document template more efficient. Appellants assert that Claim 4 is allowable because the limitations of Claim 4, the Adata file and substantial semantic equivalence, are not disclosed by Stewart, Blackman, or WIDL, as asserted by the Examiner.

Give all claim limitations proper weight

[0071] In the present case, Appellants amended Claims 4, 14, and 24 to clarify what is meant by a transaction processing system message definition. Specifically, the Appellants added the phrase "substantially semantically equivalent to" to Claims 4, 14, and 24. Appellants respectfully assert that the Examiner failed to give proper consideration to the added clarifying phrase. Therefore, the Examiner failed to consider the "entirety of the claimed invention."

[0072] "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). *MPEP* §2143.03. Although the meaning of the phrases were not clear to the Examiner, which prompted the rejection under 35 U.S.C §112, 2nd paragraph, the clarifying language clearly

distinguishes the claims from the art of record.

[0073] For reasons explained above, those of skill in the art will recognize that the transaction processing system message definition defines the syntax and semantics. Furthermore, the meaning (semantics) of data in the messages that are exchanged is critical to successful messaging between the client and client terminals and the transaction processing system such as IMS. As indicated above, these amendments are supported in the specification and should have been considered in determining the obviousness of these claims in relation to the prior art of record.

[0074] Appellants respectfully assert that this oversight by the Examiner has cost the Appellants unnecessary delay and expense in procuring a patent for the claimed invention. As the features included in the amended statement bolster Appellants' assertion that Claims 4, 14, and 24 are nonobvious. Appellants respectfully request that the rejection be overruled.

Claim 5.

[0075] Claim 5 depends from Claim 1 and should be allowed for all the same reasons stated above regarding Claim 1. In addition, Claim 5 teaches "extracting the transaction processing system message definition from one of an application source code file and a copy file." *Claims*, 5. The Examiner cites Blackman in support of this teaching. *Final* Office Action, page 9. The Examiner seems to equate a record layout with a transaction processing system message definition. However, Appellants assert a clear difference between a record layout and a transaction processing system message definition.

[0076] As indicated above, those of skill in the art recognize the difference between an interprocess message defined by a transaction processing system message definition and a database record. The difference relates to how the messages are used. For example, input/output messages may include much more than just data fields passed to a data store. For example, input/output messages defined by transaction processing system message definitions may include display, output and formatting information. In contrast, fields in a database record relate solely to data that is stored in the database. There is no display or formatting information included. The record is a simple data structure holding data. Formatting information may be particularly

important when the client of the transaction processing system is a terminal or terminal emulator. Therefore, Blackman fails to teach or disclose the subject matter of Claim 5 because Blackman relates solely to database record formats and not transaction processing system message definitions.

[0077] Thus, Stewart, Blackman, and WIDL fail to teach or suggest all the limitations recited in the Claims 1, 2, 3, 4, and 5. Claims 11-15 and 21-25 depend from these claims. Therefore, Appellants respectfully submit that the rejection of Claims 1-5, 11-15, and 21-25 be overruled.

III. The rejection under 35 U.S.C. §103(a) of Claims 6, 16, and 26 as obvious in view of Stewart, Blackman, WIDL, Lyengar, and Brodsky.

[0078] The Prior Art. The Stewart, Blackman, and WIDL are summarized above. Lyengar and Brodsky are summarized below.

[0079] Lyengar and Brodsky. Lyengar teaches transformation of distinct business models into a generic representation format, such as UML. *See* Lyengar Abstract. Brodsky discusses benefits and advantages to XML Metadata Interchange (XMI). *See* Brodsky page 1.

Claim 6.

[0080] Claim 6 depends from Claim 2 and should be allowed for all the same reasons stated above regarding Claim 2. In addition, claim 6 adds the limitation of "**creating a UML object model** for representing **arbitrary** transaction processing system message definitions; and processing the object model using an XMI utility to generate the DTD." Claims 16 and 26 recite substantially the same subject matter as Claim 6. The UML object model allows for a more generic representation of transaction processing system message definitions, specifically arbitrary transaction processing system message definitions. *See* specification page 22, lines 8-13. The UML object model defines the fields of the message definition using DataItems and a hierarchical representation. *See* specification page 22, lines 14-19, and Figure 8.

Advantageously, this allows for more message definitions to be readily used in generating the XML document template and associated XML document which allows for messages to be exchanged with a legacy transaction processing system such as IMS.

[0081] The Examiner cites Lyengar and Brodsky in support of the rejection. *Final* Office Action, page 11. Yet neither Lyengar nor Brodsky make reference to a transaction processing system message definition. Lyengar has a cursory reference to transformation of legacy business processes including legacy applications into UML format. *Lyengar* Abstract. However, there is no teaching in Lyengar that the legacy business process or legacy application is a transaction processing system message definition, much less an arbitrary transaction processing system message definition.

[0082] Appellants submit that creation of a UML object model for an arbitrary transaction processing system message definition includes more than a simple code conversion as is suggested by Lyengar. Instead, as taught in the specification, the message definition is abstracted and the hierarchical relationships between DataItems must be defined. *See* specification page 22, lines 9-13. This is particularly important where an **arbitrary** message definition is being represented. Appellants find nothing in Stewart, Blackman, Lyengar, or Brodsky to teach or suggest the step of "creating a UML object model for representing arbitrary transaction processing system message definitions" as recited in Claim 6 and supported in the specification. Appellants assert that claims 6, 16, and 26 are allowable.

IV. The rejection under 35 U.S.C. §103(a) of Claims 7-10, 17-20, and 27-30 as obvious in view of Stewart, Blackman, WIDL, and Friedman.

[0083] The Prior Art. The Stewart, Blackman, and WIDL are summarized above. Friedman is summarized below.

[0084] Friedman. Friedman teaches the relationship between a DTD and an XML document. *See* Friedman Col. 12, lines 14-20, Abstract. Specifically, Friedman teaches extraction of information from natural-language text data. *See* Friedman Abstract.

Claim 7.

[0085] Claims 7-10 depend from Claim 2 and should be allowed for all the same reasons stated above regarding Claim 2. In addition, Claim 7 further defines the merging step and "identif[ies] a placeholder within the XML document template... read[s] the value from the transaction processing system message; and insert[s] the value into a location within the XML document template indicated by the placeholder." Claims 17 and 27 recite substantially the same subject matter. Appellants have explained the lack of teaching in the art of record for an XML document template and transaction processing system message. Claims 7-10 relate several additional claimed steps that are neither, taught, disclosed, or suggested in the prior art of record.

[0086] Claim 7 relates a process of using an XML document template in creating an XML document. Appellants find no teaching in Blackman or WIDL to generate a template in XML or any other format based on Input/Output message formats, transaction processing system message definitions. The definition of template cited by the Examiner in the first office action (See page 8) omits one important characteristic of a template. A template leaves openings for data to be inserted. www.webopedia.com includes the following definition of template "In spreadsheet and database applications, a template is a blank form that shows which fields exist, their locations, and their length. In spreadsheet applications, for example, a template is a spreadsheet in which **all** the **cells** have been **defined but no data** has yet been **entered**." (Emphasis Added).

[0087] The openings for data in a template are indicated by placeholders as recited in Claim 7. Data values are read from the transaction processing system message and placed in the XML document template at locations indicated by the placeholders. Reading of data from the transaction processing system message is not a trivial exercise. *See* specification page 26, lines 18-21. In fact, due to the size dependent nature of the transaction processing system messages and message fields, Claim 10 further defines placeholders that include size indicators. This is done so that data fields of the appropriate sizes in the transaction processing system message can be read and placed in the XML document template. *See* specification page 27, lines 3-9. Once all the data from a transaction processing system message is inserted, the XML template becomes an XML document suitable for open exchange with clients and terminals configured to interpret and use an XML document. *See* specification page 7, lines 1-6, and page 26, lines 1-5.

[0088] Appellants find no teaching or suggestion in Friedman, Blackman, or WIDL that a template is created, or that the template is filled in with data from a legacy data structure such as a transaction processing system message. Instead, Friedman teaches the relationship between a DTD and an XML document. *Friedman*, col. 12, lines 14-17. However, the relationship between an XML document and a DTD is not the same as that of a template (an XML document template) and a data structure (a transaction processing system message) holding data to be placed in the template.

[0089] A DTD indicates the syntax, or format, for XML documents associated with the DTD. Specifically, a DTD sets out the names for different types of XML elements, where these can occur, and how they relate to other XML elements. *See* www.ucc.ie/xml/faq.html. Consequently, DTD are used to verify that XML documents are properly formed. *Friedman*, col. 12, lines 14-17. The DTD indicates whether or not an XML document contains the expected fields organized in the expected manner. The DTD contains no placeholders indicating where data from a source data structure such as a transaction processing system message is to be placed in an XML document template. Without placeholders, a computer software process has no way to identify where data in fields of a transaction processing system message is to be placed in an XML document **template**. A clear understanding of a template is important to understanding how the relationship between XML document templates and transaction processing system messages differs greatly from the relationship of DTDs to XML **documents** taught in Friedman. Any artisan of skill in the art of computer science recognizes a difference between an XML document template and an XML document.

[0090] Appellants submit that Friedman fails to teach, disclose, or suggest processing of an XML document template using a transaction processing system message to generate an XML document by replacing placeholders with the proper data in the message as recited in Claims 7-10. The teachings of Friedman and the other art of record have nothing to do with document templates, transaction processing system messages, and placeholders. Therefore, Appellants request that the rejection of Claims 7-10, 17-20, and 27-30 be withdrawn.

8. CLAIMS APPENDIX

Claims involved in the appeal

1. (Previously Amended) A computer-implemented method for representing transaction processing system messages as XML documents, the method comprising:

generating an XML document template from a transaction processing system message definition, the message definition representative of the syntax and semantics for messages exchanged with the transaction processing system; and

merging a transaction processing system message with the generated template to produce a corresponding XML document.

2. (Previously Amended) The method of claim 1, wherein the generating step comprises:

obtaining a transaction processing system message definition;

obtaining a DTD for representing arbitrary transaction processing system message definitions;

compiling the transaction processing system message definition with an option configured to produce an associated data (Adata) file; and

parsing the Adata file using the DTD to generate an XML document template corresponding to the transaction processing system message definition.

3. (Previously Amended) The method of claim 1, wherein the generating step comprises:

obtaining a transaction processing system message definition;

obtaining a DTD for representing arbitrary transaction processing system message definitions; and

parsing the transaction processing system message definition using the DTD to generate an XML document template corresponding to the transaction processing system message definition.

4. (Previously Amended) The method of claim 2, wherein the Adata file comprises a transaction processing system message definition in a format substantially semantically equivalent to program source code from which the transaction processing system message definition originates.

5. (Previously Amended) The method of claim 2, wherein obtaining the transaction processing system message definition comprises:

extracting the transaction processing system message definition from one of an application source code file and a copy file.

6. (Previously Amended) The method of claim 2, wherein the step of obtaining the DTD comprises:

creating a UML object model for representing arbitrary transaction processing system message definitions; and

processing the object model using an XMI utility to generate the DTD.

7. (Previously Amended) The method of claim 2, wherein the merging step comprises:

identifying a placeholder within the XML document template for receiving a corresponding value from the transaction processing system message;

reading the value from the transaction processing system message; and

inserting the value into a location within the XML document template indicated by the placeholder.

8. (Original) The method of claim 7, wherein the placeholder comprises an XML tag.

9. (Previously Amended) The method of claim 7, wherein the identifying step comprises:

checking the placeholder for an associated tag indicating that a corresponding value exists within the transaction processing system message.

10. (Previously Amended) The method of claim 7, wherein the placeholder has an associated tag indicating the size of the corresponding value within the transaction processing system message, the reading step comprising:

reading a portion of the transaction processing system message corresponding to the indicated size.

11. (Previously Amended) A system for representing transaction processing system messages as XML documents, the system comprising:

a template generation module configured to generate an XML document template from a transaction processing system message definition, the message definition representative of the syntax and semantics for messages exchanged with the transaction processing system; and

a merging module configured to merge a transaction processing system message with the generated template to produce a corresponding XML document.

12. (Previously Amended) The system of claim 11, wherein the template generating module comprises:

a compiler configured to compile a transaction processing system message definition with an option configured to produce an associated data (Adata) file; and

a parser configured to parse the Adata file using a DTD for representing arbitrary transaction processing system message definitions to generate an XML document template corresponding to the transaction processing system message definition.

13. (Previously Amended) The system of claim 11, wherein the template generating module comprises:

a parser configured to obtain a DTD for representing arbitrary transaction processing system message definitions and parse the transaction

processing system message definition using the DTD to generate an XML document template corresponding to the transaction processing system message definition.

14. (Previously Amended) The system of claim 12, wherein the Adata file comprises a transaction processing system message definition in a format substantially semantically equivalent to program source code from which the transaction processing system message definition originates.

15. (Previously Amended) The system of claim 12, further comprising:

a message definition extractor configured to extract the transaction processing system message definition from one of an application source code file and a copy file.

16. (Previously Amended) The system of claim 12, further comprising:

a modeling tool configured to create a UML object model for representing arbitrary transaction processing system message definitions; and

an XMI utility for generating the DTD from the UML object model.

17. (Previously Amended) The system of claim 12, wherein the merging module is further configured to identify a placeholder within XML document template for receiving a corresponding value from the transaction processing system message; read the value from the transaction processing system message; and insert the value into a location within the XML document template indicated by the placeholder.

18. (Original) The system of claim 17, wherein the placeholder comprises an XML tag.

19. (Previously Amended) The system of claim 17, wherein the placeholder comprises an associated tag indicating whether a corresponding value exists within the transaction processing system message.

20. (Previously Amended) The system of claim 17, wherein the placeholder has an associated tag indicating the size of the corresponding value within the transaction processing system message.

21. (Previously Amended) An article of manufacture comprising a program storage medium readable by a processor and embodying one or more instructions executable by the processor to perform a computer-implemented method for representing transaction processing system messages as XML documents, the method comprising:

generating an XML document template from a transaction processing system message definition, the message definition representative of the syntax and semantics for messages exchanged with the transaction processing system; and

merging a transaction processing system message with the generated template to produce a corresponding XML document.

22. (Previously Amended) The article of claim 21, wherein the generating step comprises:

obtaining a transaction processing system message definition;

obtaining a DTD for representing arbitrary transaction processing system message definitions;

compiling the transaction processing system message definition with an option configured to produce an associated data (Adata) file; and

parsing the Adata file using the DTD to generate an XML document template corresponding to the transaction processing system message definition.

23. (Previously Amended) The article of claim 22, wherein the transaction processing system message definition comprises program source code in a language selected from the group consisting of COBOL, PL/I, Assembler, and Pascal.

24. (Previously Amended) The article of claim 22, wherein the Adata file comprises a transaction processing system message definition in a format substantially semantically equivalent to program source code from which the transaction processing system message definition originates.

25. (Previously Amended) The article of claim 22, wherein obtaining the transaction processing system message definition comprises:

extracting the transaction processing system message definition from one of an application source code file and a copy file.

26. (Previously Amended) The article of claim 22, wherein the step of obtaining the DTD comprises:

creating a UML object model for representing arbitrary transaction processing system message definitions; and

processing the object model using an XMI utility to generate the DTD.

27. (Previously Amended) The article of claim 22, wherein the merging step comprises:

identifying a placeholder within XML document template for receiving a corresponding value from the transaction processing system message;

reading the value from the transaction processing system message; and

inserting the value into a location within the XML document template indicated by the placeholder.

28. (Original) The article of claim 27, wherein the placeholder comprises an XML tag.

29. (Previously Amended) The article of claim 27, wherein the identifying step comprises:

checking the placeholder for an associated tag indicating that a corresponding value exists within the transaction processing system message.

30. (Previously Amended) The article of claim 27, wherein the placeholder has an associated tag indicating the size of the corresponding value within the transaction processing system message, the reading step comprising:

reading a portion of the transaction processing system message corresponding to the indicated size.

9. EVIDENCE APPENDIX

In compliance with CFR §41.12 the non-binding authorities relied upon above are cited below. Copies of these references are also included.

Internet word search for term "syntax" on the website at URL http://www.webopedia.com

Internet word search for term "semantics" on the website at URL http:// www.webopedia.com

Internet word search for term "template" on the website at URL http://www.webopedia.com

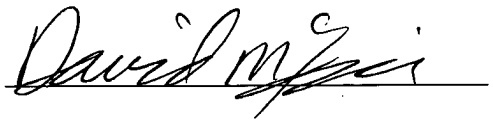
10. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.

SUMMARY

[0091] In view of the foregoing, the Examiner has not properly established *prima facie* obviousness of claims 1-30. Appellants respectfully request reversal of the Section 103 rejection and allowance of claims 1-30. Appellants submit that the foregoing arguments further establish the non-obviousness of the present invention. Reversal of the rejections and allowance of the appealed claims is respectfully requested.

Respectfully submitted,



David J. McKenzie
Reg. No. 46,919
Attorney for Applicant

Date: December 2, 2004
8 East Broadway, Suite 600
Salt Lake City, UT 84111
Telephone (801) 994-4646
Fax (801) 322-1054

MENU
Home
Term of the Day
New Terms
Pronunciation
New Links
Quick Reference
Did You Know?
Search Tool
Tech Support
Webopedia Jobs
About Us
Link to Us
Advertising

Compare Prices:
go
HardwareCentral

Talk To Us...
Submit a URL
Suggest a Term
Report an Error

YAHOO! shopping

internet.com
Developer
Downloads
International
Internet Lists
Internet News
Internet Resources
IT
Linux/Open Source
Small Business
Windows Technology
Wireless Internet
xSP Resources

Search internet.com

semantics

Last modified: Sunday, September 01, 1996

In linguistics, the study of meanings. In computer science, the term is frequently used to differentiate the meaning of an instruction from its format. The format, which covers the spelling of language components and the rules controlling how components are combined, is called the language's *syntax*. For example, if you misspell a command, it is a syntax error. If, on the other hand, you enter a legal command that does not make any sense in the current context, it is a semantic error.

•E-mail this definition to a colleague•

*For internet.com pages about **semantics** **CLICK HERE**. Also check out the following links!*

LINKS = Great Page!

Related Categories
Programming

Related Terms
parse
parser
programming language
syntax

(Webopedia)
Give Us Your Feedback

MENU
Home
Term of the Day
New Terms
Pronunciation
New Links
Quick Reference
Did You Know?
Search Tool
Tech Support
Webopedia Jobs
About Us
Link to Us
Advertising

Compare Prices: go
HardwareCentral

Talk To Us...
Submit a URL
Suggest a Term
Report an Error

Travel Ideas:
Orlando Hotels
San Diego Hotels
Seattle Hotels
Fort Lauderdale
St Augustine
Disney Vacations
Discount Travel
French Quarter

internet.com
Developer
Downloads
International
Internet Lists
Internet News
Internet Resources
IT

syntax

Last modified: Friday, September 26, 2003

Refers to the spelling and grammar of a programming language. Computers are inflexible machines that understand what you type only if you type it in the exact form that the computer expects. The expected form is called the syntax.

Each program defines its own syntactical rules that control which words the computer understands, which combinations of words are meaningful, and what punctuation is necessary.

•E-mail this definition to a colleague•

*For internet.com pages about **syntax** **CLICK HERE**. Also check out the following links!*

LINKS = Great Page!

Related Categories
Programming Languages

Related Terms
language
programming language
semantics

(Webopedia)
Give Us Your Feedback

Flat Panel Televisions
syntax Products

MENU
Home
Term of the Day
New Terms
Pronunciation
New Links
Quick Reference
Did You Know?
Search Tool
Tech Support
Webopedia Jobs
About Us
Link to Us
Advertising

Compare Prices: go
HardwareCentral

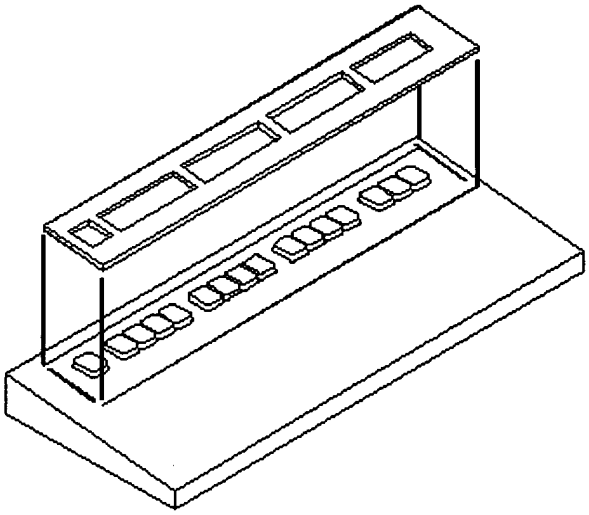
Talk To Us...
Submit a URL
Suggest a Term
Report an Error

Travel Ideas:
Kauai Hawaii
Los Angeles Hotels
Myrtle Beach
Fort Lauderdale
Spring Break
Disney World
BookIt.com
New Orleans Hotels

internet.com
Developer
Downloads
International
Internet Lists
Internet News
Internet Resources
IT

template

Last modified: Wednesday, April 07, 2004



(n.) *(1)* Something that establishes or serves as a pattern for reference.

(2) A plastic or paper diagram that you can put on your keyboard to indicate the meanings of different keys for a particular program.

(3) A sheet of plastic with menus and command boxes drawn on it that you place on top of a digitizing tablet. You can select commands by pressing the digitizing tablet's pen against a command box or by positioning the cursor over a box and pressing one of the cursor keys.

(4) In spreadsheet and database applications, a template is a blank form that shows which fields exist, their locations, and their length. In spreadsheet applications, for example, a template is a spreadsheet in which all the cells have been defined but no data has yet been

entered.

(5) In some word processing applications, *template* is used in place of *style sheet*.

(6) DOS uses the term *template* to mean command buffer.

(7) A shortened term for a biometric *reference template*.

•E-mail this definition to a colleague•

*For internet.com pages about **template** CLICK HERE. Also check out the following links!*

LINKS = Great Page!

Related Categories

Data

Desktop Publishing

Related Terms

boilerplate

command buffer

cursor

digitizing tablet

(Webopedia)
Give Us Your Feedback